The predetermined hyperlink address may comprise a local storage address or an Internet Web address. Said hyperlink address string may comprise a character string and said predetermined hyperlink address may comprise an URL or URI.

Said hyperlink address string may further include a third attribute indicating said means for hyperlinking is user-activated and further comprising: means for storing, means for visually displaying and means for user-activating, each operably coupled to said means for processing and collectively configured for user-activating said means for hyperlinking to hyperlink to said predetermined hyperlinked content, said means for storing to store for predetermined time said predetermined hyperlink address and said means for visually displaying to visually display for predetermined time predetermined data to prompt said user-activating said means for hyperlinking to hyperlink to said predetermined hyperlinked content, both said means for storing and said means for visually displaying responsive to said third attribute and said means for user-activating to activate said means for hyperlinking to hyperlink to said predetermined hyperlinked content. Said hyperlink address string may further include a third attribute indicating said means for hyperlinking is automatically activated and further comprising means for automatically activating said means for hyperlinking operably coupled to said means for processing and configured for automatically activating said means for hyperlinking to automatically hyperlink to said predetermined hyperlinked content responsive to said third attribute.

Said one or a plurality of predetermined parameters defining said pre-defined printable output of said predetermined hyperlinked content may comprise at least one parameter specifying predetermined print activation for activating printing of said pre-defined printable output of said predetermined hyperlinked content. Said one or a plurality of predetermined parameters defining said pre-defined printable output of said predetermined hyperlinked content may comprise at least one parameter specifying a predetermined spatial or temporal aspect of said predetermined hyperlinked content.

Referring to the FIG. **1**, said program signal receiver apparatus may comprise a television receiver apparatus, a radio receiver apparatus and/or a media player apparatus. Said program signal receiving means may comprise tuner means, i.e. TELEVISION TUNER **22**. TELEVISION TUNER **22** may receive said URL string transmitted via TELEVISION DATA TRANSMISSION LINK **21**. Said program signal receiving means **22** may comprise any video or audio program receiver means such as broadcast, cable or satellite television or radio tuner means, set-top box means, Internet program signal receiver means or other program signal receiver means. In another embodiment, said program signal receiving means **22** may comprise storage medium means such as video or audio recorder or player means.

Via DATA INPUT/OUTPUT **23**, said URL string may be input to MICROPROCESSOR **24** (which may comprise any conventional data processor, microprocessor, central processing unit or equivalent data processing means). ROM **25** may store the program of instructions which controls MICROPROCESSOR **24**. Responsive to <text>Advertiser Coupon</text> 9A, MICROPROCESSOR **24** may route URL from <area href="http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072" shape="rectangle" coords="10,20,40,60"/> 6A or <area href="http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072" shape="rectangle" coords="10,20,40,60"/> 6B via BUS **27** to be temporarily stored in RAM **28**, route <text>Advertiser Coupon</text> 9A via INTERFACE DEVICE **29** to be displayed as non-hypertext or, optionally, "hyperlink text" via LCD **30** and detect the URL expiration date/time from begin="indefinite" dur="media" 7A. The term "hyperlink text" as used herein encompasses the meaning including hypertext conferred thereon in Boden, et al, U.S. Pat. No. 5,930,512, Nielsen, U.S. Pat. No. 6,199,071, Carroll, et al, U.S. Pat. No. 6,154,205 and Kato, U.S. Pat. No. 5,809,512, which are incorporated herein by reference.

Said apparatus may further comprise: time measuring means, i.e. DATE/TIME CLOCK **26**, operably coupled to said data processing means MICROPROCESSOR **24** for said data processing means MICROPROCESSOR **24** to determine when a time referenced by an attribute included in said hyperlink address string transpires. In one preferred embodiment, said URL expiration date/time attribute **7** may be detected and compared by comparing means with current date/time information from DATE/TIME CLOCK **26** first before one or more other attributes in said URL string is processed to ascertain first if said URL remains valid. DATE/TIME CLOCK **26** and/or said comparing means may be remote from or built-into said apparatus such as disclosed in Danneels, U.S. Pat. No. 5,602,992 or Maturi, et al, U.S. Pat. No. 5,559,999, which are incorporated herein by reference.

Said apparatus may further comprise: memory means RAM **28** and visual display means LCD **30**, both operably coupled to said data processing means MICROPROCESSOR **24** and collectively configured for said user-activating said hyperlink to said predetermined hyperlink address responsive to said user-activating attribute <text>Advertiser Coupon</text> 9A, said memory means RAM **28** for storing for predetermined time said predetermined hyperlink address specified in said first attribute <area href="http://PrintHD.TV/text-2/ad/ad2.aspHD=7072" shape="rectangle" coords="10,20,40,60"/> 6A and said visual display means LCD **30** for visually displaying, optionally with a text blinking effects attribute, for predetermined time predetermined data <text>Advertiser Coupon</text> 9A to prompt said user-activating said hyperlink to said predetermined hyperlink address wherein <text>Advertiser Coupon</text> 9A serves as both said user-activating attribute and predetermined data for display, however, said predetermined data for display may be pre-stored as text or a graphic image in predetermined storage means in said apparatus, and user-activating means, i.e. PUSHBUTTON **31**, operably coupled to said data processing means MICROPROCESSOR **24** for said user-activating said hyperlink to said predetermined hyperlink address.

If user-activating attribute <text>Advertiser Coupon</text> 9A is received in said URL string, an hyperlink to WEB PAGE **35** may be user-activated via PUSHBUTTON **31**. Visual display means such as LCD **30** may be used to alert an user that user activation of said hyperlink is available to be made. LCD **30**, for example, may visually display "[Advertiser Name] Coupon" as text or, optionally, hypertext from <text>Advertiser Coupon</text> 9A to inform viewers that an hyperlink to coupon WEB PAGE **35** is available. In addition to indicating when program-related URL in <area href="http://PrintHD.TV/text-2/ad/ad2.asp?B)=7072" shape="rectangle" coords="10,20,40,60"/> 6A is no longer valid, begin="indefinite" dur="media" 7A may specify a stop display time as well so that said display coincides with the time when coupon WEB PAGE **35** is available to be accessed.

LED or equivalent visual display means rather than LCD **30** may be used as said visual display means. In addition, user activation means other than PUSHBUTTON **31** may be used such as keypad or VRU. Activating PUSHBUTTON **31**, directly or via remote control, may send a control signal to MICROPROCESSOR **24**, which, in turn, may instruct RAM **28** to output the stored URL **6** via BUS **32** to MEMORY OUTPUT **33**.

Both LCD **30** and PUSHBUTTON **31** may be built into a conventional remote control unit and communicably coupled to MICROPROCESSOR **24**. INTERFACE DEVICE **29** may then connect to a conventional two-way infrared (IR) link coupled to said remote control unit to send and receive control signals.

Responsive to actuate="on Load" 9B, MICROPROCESSOR **24** may route the URL from <area href="http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072" shape="rectangle" coords="10,20,40,60"> 6B via BUS **27** to LINK CONTROLLER **34** to automatically activate the URL hyperlink to WEB PAGE **36** at begin="dd:hh:mm:ss.0.sup.1-n" (NOT SHOWN). In addition, MICROPROCESSOR **24** may detect the URL expiration date/time from end="dd:hh:mm:ss.0.sup.1-n" (NOT SHOWN).

Upon said user-activated or automatic activation, an hyperlink to WEB PAGE **36** may be established via WEB BROWSER/PRINTER **35**. Hyperlinked content may then be displayed and printed via PrintWriter **35** on TELEVISION SCREEN (NOT SHOWN) at the specified shape="rectangle" coords="10,20,40,60" location. COUPON **37** may then be printed without user activation pursuant to, for example, an HTML instruction included in said URL string, via WEB BROWSER/PRINTER **35**. If not automatic, once images are displayed on the screen, the user may press PRINT BUTTON (NOT SHOWN) at WEB BROWSER/PRINTER **35** to commence printing in order to print a certain displayed image.

In addition to automatic hyperlink activator, LINK CONTROLLER **34** may also serve as an hyperlink de-activator. For example, LINK CONTROLLER **34** may automatically de-activate an advertisement-related hyperlink to an advertiser coupon WEB PAGE **35** when the advertisement ends. In said embodiment, begin="indefinite" dur="media" 7A or begin="dd:hh:mm:ss.0.sup.1-n" end="dd:hh:mm:ss.0.sup.1-n" (NOT SHOWN) may reference an automatic hyperlink de-activate HTML or XML instruction.

A. Print to Paper Printing

First Exemplary Embodiment

One embodiment of the invention entails printing content to paper using an printer attached to a program receiver. Web browsers may be connected to program receivers for added functionality including hyperlinking and printing means that are inherent in web browser technology.

Print data may be transmitted with the program signal or referenced at a remote location such as on the Internet. XHTML-Print may be used by the hyperlinking and printing means to print the pre-defined printable output of the predetermined hyperlinked content. In XHTML-Print, the audio/video program signal may carry the extra information for printing referenced content wherein a tag may correlate the printable content with the corresponding broadcast program content.

Content to be printed may be transmitted in a transport stream via an object carousel along with corresponding program material. Such content to be printed may be described by a Link_Descriptor( ) pointing to predetermined content located at a transport stream URL. In such system using the Link_Descriptor( ) hyperlinking is accomplished via a link table containing the Link_Descriptor( ) for each program element having a hyperlink wherein the ID field of the link table uniquely identifies a destination page to which the program element is linked and a program ID field contains the program ID of the program that carries the destination page. The carousel may transmit **10** program elements in each program with 9 containing pages and 1, the first, containing metadata describing the other 9 pages. Each group of 9 pages may be image pages corresponding to the program mapped to a digital channel for access. To display a page, the STB browser tunes to the appropriate digital channel and selects the appropriate program element (service component). The browser also selects metadata from the first page to execute the link. A maximum of 6,700 page images may be carried by one carousel.

The print to paper embodiment of the invention is demonstrated in FIG. **2**. This print to paper embodiment of the invention entails printing to paper using, preferably, the web browser printer. The hyperlink address string may then be designed to automatically invoke the browser interpreting the second attribute defining the pre-defined printable output of hyperlinked content using script to send such printable output to the printer. For demonstrative purposes only, the predetermined hyperlink address in the generated URL String, for the first exemplary embodiment (print to paper) disclosed herein, is <area href="http://PrintHD.TV/text-2/ad/ad2.aspHD=7072" shape="rectangle" coords="10,20,40,60" 6A.

Print ( ) attribute. The Print ( ) attribute may comprise at least one parameter defining the pre-defined printable output of the predetermined hyperlinked content. Thereby, this attribute may define such pre-defined printable output of the predetermined hyperlinked content. In FIG. **2**, this attribute has the value, for demonstrative purposes only, of on Select="self.print( )" 11A, indicating to send the content in focus, such as during an advertisement an advertiser coupon located at http://PrintHD.TV/text-2/ad/ad2.aspHD=7072 6A, a file hyperlinked to via an object carousel at a transport stream URL or a frame currently being displayed, to the printer. The script "self.print( )" defines only pre-defined printable output of hyperlinked content such as the coupon, file or frame currently in focus to send to printer.

The first attribute to hyperlinked content and second attribute defining the printable output may include any appropriate parameter for defining printable output such as to print an image iframe img src="images/display/printer.gif" width="24" alt=" " title="Print"onclick="printPic( )"/ wherein printPIc( ) calls window.open(url, name, options) or to print an iframe <iframe src="page2print.htm" name="ifr" height="0" width="0" frameborder="0" onClick="frames ['ifr'].print( )"></iframe>. Changing onClick to on Load will cause the print window dialogue box to pop-up when the page loads. Other exemplary embodiments of such hyperlink address string may comprise img src="page2print.htm", txt src="page2print.txt", videosrc="page2print.mpg" or, with reference to a map such as <map name="pages2print.Map">, <area href="page2print.htm" shape=" . . . " coords=" . . . ">, and rather than onClick="self.print( )" 11B said second attribute as onClick="printFile( )", onClick="printUri( )", onClick="printScreen( )" or onClick="printTemplate( )".

The hyperlink address string disclosed herein could combine hyperlinking and printing within the same first and second attributes in the string such as: Click to Print This Page, consisting of an applet to send the displayed content to a separate window for printing wherein window.close( ); may be added to automatically close the print dialogue window after printing. In particular, the window:print( ) applet is both hyperlinked to and pre-defines the content to be printed. In said string structure, "pagePrint ([URL])" or "framePrint([frame name or ID])" may replace "window.print( )" to define the printable output. An alternative structure of said first attribute and second attribute may be

<a onClick="window.print( );". Another embodiment of said first and/or second attributes in said hyperlink address string may include <a onClick="document.getElementByID(' [Content ID]') printWithDialog( )".

The attribute class="Print" may comprise said second attribute defining printable output of hyperlinked content referenced in said first attribute. Said attribute may characterize content located at a block or division or an individual cell in a table in a document as printable content.

In said print to paper embodiment of the invention, the hyperlinking and printing means may comprise means for acquiring print data included in broadcast waves such as via a transport stream in an object carousel and requesting the print data be printed to paper, means for capturing data broadcast images that are being displayed and requesting the data broadcast images be printed to paper and means for accessing print data stored in servers on the Internet and requesting said print data be printed to paper.

B. Print to File Printing

Second Exemplary Embodiment

This embodiment of the invention is described in terms of a cable television system implementation of the invention. It is understood that the scope of the invention applies to equivalent systems in broadcast or satellite television or radio implementations of the invention.

Digital cable television technology is characterized by set top box (STB) receivers receiving program signal transmissions from a local head-end. In accessing the transmissions, STB receivers may connect with a servlet located at a predetermined URL, a hyperlink address, to access a given web page corresponding to cable television program material. Once connected to the servlet, a PrintWriter class may be activated to print to the STB screen the web page corresponding to the program material.

The second embodiment of the invention is a print to file application preferably using PrintWriter or equivalent means to process and display (print) content on a screen or to a file or OutputStream. As demonstrated in the FIG. **3**, this print to file application may entail processing one or more PrintWriter attributes to print text data to a character stream, PrintWriter writer=response.getWriter( ) 7B, writer.println("</body></html>"); 8B and/or ServletOutputStream out=response.getOutputStream( ); 9B along with the attribute 6B indicating the predetermined hyperlink address to the predetermined hyperlinked content. In FIG. **3**, said predetermined hyperlink address is, for demonstrative purposes only, http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072 6B, a web page coupon.

A servlet element has dynamic properties representing the servlet parameters, which are defined using a servlet parameter declaration. Servlet parameters are generated and transmitted in the servlet-STB client connection. Said parameters specify parameters for STB hyperlink access to the servlet. The one or a plurality of predetermined parameters defining the pre-defined printable output of the predetermined hyperlinked content indicated in the second attribute may include one or more servlet parameters.

The PrintWriter also has parameters for various methods defined in the Java programming language. PrintWriter parameters for using methods in java.awt include:
Component.list(PrintWriter out) to print a listing to a specified print writer;
Component.list(PrintWriter out, int indent) to print out a list, starting at the specified indention, to a specified print writer or Container.list(PrintWriter out, int indent) to print out a list, starting at the specified indention, to the specified print writer.
PrintWriter parameters using methods in java.lang include:
Throwable.printStackTrace(PrintWriter s) to print this Throwable and its backtrace to a specified print writer;
ExceptionInInitializerError.printStackTrace(PrintWriter pw) to print the stack trace of the exception that occurred to the specified print writer or
ClassNotFoundException.printStackTrace(PrintWriter pw) to print the stack backtrace to a specified print writer. PrintWriter parameters using methods in java.lang.reflect include:
InvocationTargetException.printStackTrace(PrintWriter pw) to print the stack trace of the thrown target exception to a specified print writer or
UndeclaredThrowableException.printStackTrace(PrintWriter pw) to print this UndeclaredThrowableException and its backtrace to a specified PrintWriter.
A PrintWriter parameter using methods in java.rmi includes:
RemoteException.printStackTrace(PrintWriter pw) to print the composite message and the embedded stack trace to a specified print writer pw. A PrintWriter parameter using methods in java.rmi.activation includes:
ActivationException.printStackTrace(PrintWriter pw) to print the composite message and the embedded stack trace to the specified print writer pw. A PrintWriter parameter using methods in java.rmi.server includes:
ServerCloneException.printStackTrace(PrintWriter pw) to print the composite message and the embedded stack trace to the specified writer pw. A PrintWriter parameter using methods in java.security includes:
PrivilegedActionException.printStackTrace(PrintWriter pw) to print the stack trace of the exception that occurred to the specified print writer. A PrintWriter parameter using methods in java.sql includes: DriverManager.getLogWriter( ). A PrintWriter parameter using methods in java.sql includes:
DriverManager.setLogWriter(PrintWriter out) to set the logging/tracing PrintWriter object that is used by the DriverManager and all drivers.
A PrintWriter parameter using methods in java.util includes:
Properties.list(PrintWriter out) to print a property list out to a specified output stream. A PrintWriter parameter using methods in javax.naming includes:
NamingException.printStackTrace(PrintWriter pw) to print this exception's stack trace to a print writer. The one or a plurality of predetermined parameters defining the pre-defined printable output of the predetermined hyperlinked content indicated in the second attribute may include one or more PrintWriter parameters.

The server computer passes objects that implement the "HttpRequest", while the servlet returns an "HttpResponse" object. The "ServletContext" interface is used to exchange information with the server environment. Some of the methods on the "ServletContext" object are "Getserver( )" and "GetServlets( )". "GetServer" returns a pointer to the parent server within which the instantiated Httpservlet runs. Using this pointer, the HttpServlet object can find out information about its parent server. The "GetServlet" method returns pointers to the servlets running on the parent server. The "ServerProperties" interface is used to exchange information regarding specific server properties established by a server administrator.

The service method is provided with Request and Response parameters included in the one or more predetermined parameters defining the pre-defined printable output of the predetermined hyperlinked content indicated in the second attribute in the hyperlink address string. These parameters encapsulate the data sent by the client, thereby allowing

servlets to report status information such as errors. Servlets normally retrieve most of their parameters through an input stream, and send their responses using an output stream: ServletInputStream in=request.getInputStream ( ): ServletOutputStream out=response.getOutputStream ( ); These input and output streams may be used with data in whatever format is appropriate. For example, an applet and servlet might exchange data using object serialization, HTML, or any number of image formats.

Since servlets are JAVA objects, they have instance-specific data. This means that in effect servlets are independent applications running within servers, without needing the complexity of additional classes (which are required by some alternative server extension APIs). Servlets have access to some servlet-specific configuration data at initialization time. This allows different instances of the same servlet class to be initialized with different data, and be managed as differently named servlets. The data provided at initialization time includes an area where each instance keeps its persistent instance-specific state.

In a servlet-client structure, the hyperlink address to the servlet may indicate attributes and values for printing the hyperlinked content at the servlet to the client screen or a file as in the following a sample snippet of a Servlet.

TABLE-US-00001 response.setBufferSize ("8*1024"); response.setContentSize ("text/html"); PrintWriter out = response.getWriter ( ); out.println ("<HTML>"); out.println ("<HEAD> <TITLE> Servlet Output</TITLE> </HEAD>"); out.println ("<BODY>"); out.println ("<P> Welcome To this World </P>"); out.println ("</BODY>"); out.println ("</HTML>");

In this example, the page **36** hyperlinked to is defined between the <HTML> and </HTML> tags. In one embodiment disclosed herein, this is the same page **36** that is printed to 37 (displayed on) STB client screen (NOT SHOWN).

Servlets support the familiar programming model of accepting requests and generating responses. The following is a simple servlet defining a single method called "service".

TABLE-US-00002 import java.servlet.*; public class MyServlet extends GenericServlet { public void service (ServletRequest request, ServletResponse response) throws ServletException, IOException { ... } ... }

Building upon the previous simple servlet example, the following program code is an example of a servlet that is used to send Hypertext Markup Language (HTML) text when it is invoked.

TABLE-US-00003 public class SimpleServlet extends GenericServlet { public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException { res.setContentType("text/html"); PrintWriter out = new PrintWriter(res.getOutputStream( )); out.println("<HEAD><TITLE> SimpleServlet Output </TITLE></HEAD><BODY>"); out.println("<h1> SimpleServlet Output </h1>"); out.println("<P>This is output from SimpleServlet."); out.println("</BODY>"); out.flush( ); } public String getServletInfo( ) { return "A simple servlet"; } }

The following servlet code implements asynchronous chat functionality.

```
public class ChatServlet
    extends HttpServlet implements CometProcessor  {
    protected ArrayList<HttpServletResponse> connections =
        new ArrayList<HttpServletResponse>( );
    protected MessageSender messageSender = null;
    public void init( ) throws ServletException  {
        messageSender = new MessageSender( );
        Thread messageSenderThread =
            new Thread(messageSender, "MessageSender[" +
getServletContext( ).getContextPath( ) + "]");
        messageSenderThread.setDaemon(true);
        messageSenderThread.start( );
    }
    public void destroy( )  {
        connections.clear( );
        messageSender.stop( );
        messageSender = null;
    }
    /**
     * Process the given Comet event.
     *
     * @param event The Comet event that will be processed
     * @throws IOException
     * @throws ServletException
     */
    public void event(CometEvent event)
        throws IOException, ServletException  {
        HttpServletRequest request = event.getHttpServletRequest( );
        HttpServletResponse response = event.getHttpServletResponse( );
        if (event.getEventType( ) == CometEvent.EventType.BEGIN)  {
            log("Begin for session: " +
request.getSession(true).getId( ));
            PrintWriter writer = response.getWriter( );
            writer.println("<!doctype html public \"-//w3c//dtd html
4.0 transitional//en\">");
            writer.println("<head><title>JSP Chat</title></head><body
bgcolor=\"#FFFFFF\">");
            writer.flush( );
            synchronized(connections)  {
                connections.add(response);
            }
        } else
        if (event.getEventType( ) == CometEvent.EventType.ERROR)
{
            log("Error for session: " +
request.getSession(true).getId( ));
            synchronized(connections)  {
                connections.remove(response);
            }
            event.close( );
        } else
        if (event.getEventType( ) == CometEvent.EventType.END)  {
            log("End for session: " +
request.getSession(true).getId( ));
            synchronized(connections)  {
                connections.remove(response);
            }
            PrintWriter writer = response.getWriter( );
            writer.println("</body></html>");
            event.close( );
        } else
        if (event.getEventType( ) == CometEvent.EventType.READ)  {
            InputStream is = request.getInputStream( );
            byte[ ] buf = new byte[512];
            do  {
                int n = is.read(buf); //can throw an IOException
                if (n > 0)  {
                    log("Read " + n + " bytes: " + new String(buf, 0,
n)
                            + " for session: " +
request.getSession(true).getId( ));
                } else if (n < 0)  {
                    error(event, request, response);
                    return;
                }
            } while (is.available( ) > 0);
```

-continued

```
        }
    }
    public class MessageSender implements Runnable   {
        protected boolean running = true;
        protected ArrayList<String> messages = new ArrayList<String>( );
        public MessageSender( )   {
        }
        public void stop( )   {
            running = false;
        }
        /**
         * Add message for sending.
         */
        public void send(String user, String message)   {
            synchronized (messages)   {
                messages.add("[" + user + "]: " + message);
                messages.notify( );
            }
        }
        public void run( )   {
            while (running)   {
                if (messages.size( )   ==   0)   {
                    try   {
                        synchronized (messages)   {
                            messages.wait( );
                        }
                    } catch (InterruptedException e)   {
                        // Ignore
                    }
                }
                synchronized (connections)   {
                    String[ ] pendingMessages = null;
                    synchronized (messages)   {
                        pendingMessages = messages.toArray(new
String[0]);
                        messages.clear( );
                    }
                    // Send any pending message on all the open
connections
                    for (int i = 0; i < connections.size( ); i++)   {
                        try   {
                            PrintWriter writer =
connections.get(i).getWriter( );
                            for (int j = 0; j < pendingMessages.length;
j++)   {
                                writer.println(pendingMessages[j] +
"<br>");
                            }
                            writer.flush( );
                        } catch (IOException e)   {
                            log("IOException sending message", e);
                        }
                    }
                }
            }
        }
    }
}
```

A servlet is typically instantiated on server startup. In the alternative, the servlet may be instantiated under a predetermined set of conditions or by client invocation. The servlet may be instantiated and executed by using its URL (e.g., http://hostkservlet URL>). The http protocol supports the passing of arguments, thus, arguments may be passed to the servlet (e.g., http://hostkservlet URL>?<arguments>). The properties object is a JAVA programming language properties class which comprises a set of "name:value" pairs. A system administrator can pass arguments to an instantiated HttpServlet object through the properties object. In this way, the system administrator can "customize" an HttpServlet for a particular server at a particular site. For example, the system administrator can pass the Httpservlet object site specific information about the network location of a database which stores documents that will be requested by client processes across the network or the amount of memory available in system buffers which will be used for processing the server administrator.

Java Server Pages (JSPs) are an extension to the Java servlet technology from Sun Microsystems that allows HyperText Markup Language (HTML) code to be combined with Java code on the same page. The Java provides the processing and the HTML provides the page layout that will be rendered in the Web browser. To process all JSP elements, e.g., directive elements, action elements, and scripting elements, in the JSP page, the container first turns the JSP page into a servlet (known as the JSP page implementation class). The conversion involves converting all template text to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior. The container then compiles the servlet class. The STB client hyperlinking to the page causes PrintWriter to print pre-defined printable output of hyperlinked content to STB client display screen.

In the servlet-STB client connection, the servlet is a web service interface to a library allowing dynamic MIDlets to be served over the Internet using an over-the-air protocol. The function of the servlet is to dynamically generate a MIDlet in response to specifications in a HTTP request. The HTTP request contains a URL for the requested MIDlet.

The web server hosting the servlet supports the above-noted Java Servlet Specification (Ver. 2.3 or higher). The servlet uses the web server's log file to log its operations and error states. Sessions are managed by the web server, using query string hashing. Using the same mechanism, the servlet creates user specific JAR files, preventing multiple requests from overriding one another. The one or more parameters defining the pre-defined printable output of the predetermined hyperlinked content at the servlet may define PrintWriter behavior for printing the coupon characters to the STB client screen during the playing of the associated spot.

MIDML defines a generic screen element. A screen element has an attribute "name", which is the name of the screen. An attribute "source" refers to the source MIDP-1.0/MIDML. An attribute "title" is the title of the screen. An attribute "ticker" refers to a ticker element. An attribute "persist" is a persistence flag. A screen element has an event handler "on Load", which responds to an indication to load the screen element. An event handler "onunload" responds to an indication to unload the screen element.

A form is a screen that contains an arbitrary mixture of items: images, read-only text fields, editable text fields, editable date fields, gauges, and choice group.

A TextBox is a screen that allows the user to enter and edit text. The TextBox has an attribute "maxSize", which is the maximum size of the text. An attribute "constraint" holds MIDP-1.0 text constraints, and has one of the values ANY, EMAILADDR, NUMERIC, PASSWORD, PHONENUMBER, and URL. A property "message" holds the screen's message. A property "image" represents an associated image. A property "sound" represents a sound associated with the TextBox.

A list or list class is a screen containing a list of choices. The list has an attribute "context", which holds one of the MIDP-1.0 choice context values SINGLE, and MULTIPLE. A property "choices" holds a list of choices. A property "selectedindex" holds an integer, pointing to a currently selected choice element. The value –1 indicates that no selection has been made. A property "selectedText" holds the currently selected choice text. An event "on Select" is triggered when the List-Screen choices context is updated, i.e.,

when the selected choice changes. The event "on Select" can be triggered in both SINGLE and MULTIPLE choice contexts.

Servlet Connectivity Support.

MIDML defines a servlet element to enable a developer to create interconnected applications. MIDML is thus able to exploit the Internet. It is possible, too, for the developer to specify execution of remote process, including passing of parameters and retrieving of data.

MIDML currently supports two types of servlets. The design of the servlet mechanism is generic and is reflected in the schema definition of the servlet object.

The generic servlet has an attribute "protocol", which defines the protocol used by the servlet. An attribute "http-method" has one of the values GET, POST, and HEAD, which refer to standard HTTP methods. A URL of the servlet is specified by an attribute "URL" 6B.

This generic mechanism also allows activation of a servlet with no protocol. It is possible to activate the servlet without checking the servlet response, using the value "NONE" as the protocol attribute, and the POST method as the http-method attribute.

A text servlet is a servlet, which returns simple text that can be displayed to the user using widgets. In one embodiment of the invention, a text servlet with the URL http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072 6A may generate the text for the coupon fields in INDIVIDUAL WEB PAGE FOR EACH ADVERTISER COUPON WITH COUPON FIELDS INCLUDING ADVERTISER NAME, DISCOUNT, DESCRIPTION AND EXPIRATION DATE 4. Said text may be generated using the MIDML text formatting tag and font attributes for text: font type; size (small, medium, large); style (bold, underlined, italic); face (proportional, monospace, system).

Once a servlet has been defined, it is activated using the following syntax.

```
<activateServlet servlet="sampleServlet"/>
```

Operation of the parser involves invoking a XML ParserValidator to verify the source MIDML files against schema definitions. Once the source is validated, the parser builds the project descriptor object. The project descriptor object describes a valid and parsed MIDML application using a tree structure. The tree hierarchy is identical to the original MIDML (*.midml) source code structure. The project descriptor object provides an association between the parser and the generator, and with application meta-data in the JAD file (*.jad). The project descriptor object is exposed to the developer. The project descriptor object is thus a wrapper for an entire MIDML project. It consists of two objects: an object AppMetaDescriptor, which is a JAD data wrapper, and a MIDML descriptor object model (MDOM), which is a tree oriented data structure that is disclosed in further detail hereinbelow. The descriptor object model includes elements, or readers, that can include objects that are adapted to ultimately induce generation of the following application building blocks by a code generator: MIDlet application structure; screens; widgets; events; operations; variables; and extensible elements. The objects include instance-specific data needed to generate the code. The code generation mechanism includes additional, non-instance specific data. The objects can represent multiple versions of the same building blocks that are parametrically different. For example, parametrically different screens have parameters that may include the position of a label, and the text that is displayed on the screen. These items typically vary among different screens.

The developer develops the MIDML Application and deploys it in an accessible web-server or file server. Either the developer or the end user can initiate the servlet by sending a HTTP request that is associated with the URL of the servlet. The HTPP request parameters indicate the URL for the requested MIDML application, and also includes generation directives for the compiler, so that the updated version of the MIDlet can be recompiled as appropriate.

MIDML data variables can be initialized via external parameters that are supplied once the MIDML application has been parsed and compiled. The external parameters may be supplied using a string initialization mechanism. MIDML defines a method to activate a link to an application page upon a triggered event.

MIDML defines a method to activate an arbitrary event handler upon a triggered event. MIDML defines a method to invoke Servlet calls upon a triggered event. MIDML supports page heading formats. MIDML supports line breaks within a MIDlet screen. MIDML supports a form tag as a collection of user interface widgets. The MIDML form supports a SUBMIT operation, which sends all the form's widget data to a specified link. MIDML enables the user to specify a resource location using a URL. MIDML defines a tag for a simple Boolean servlet activation. MIDML defines a tag for a text servlet activation. The library generates the MIDlet Java code from a valid MIDML starting point. The library generates the MIDlet application descriptor file (JAD). The compiler input flag is a URL or a file system path, specifying the MIDML application starting point file. The compiler has a MIDML parameters directive. The compiler has a directive to indicate packing of external classes within the generated MIDlet package (JAR).

The servlet does not generate a new MIDlet if the MIDlet requested is consistent with the cached MIDlet. MIDlet consistency is defined in terms of: file time-stamp; application descriptor information; MIDML parameters; and user agent information. This demonstrates the key importance of MIDML parameters in the servlet-STB client system. The servlet logs the following data: number of requests per MIDlet; number of created MIDlets; and number of MIDlet generation failures; and response time. Tracking of ad-coupon effectiveness is enabled via MIDlet tracking. The servlet input flag is a URL or a file system path, specifying the MIDML application starting point file. The servlet output flag is a file system path indicating the location of the resulting MIDlet JAD and JAR files. The servlet has a MIDML parameters directive.

Website www//PrintHD.TV receives the client request and determines that the client is requesting published Web page **36**. As a result, Website www//PrintHD.TV retrieves Web page **36**. The Web page **36**, being a Java Server Page, is converted into servlet http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072. Servlet http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072 includes dynamic content which is written in the Java programming language and adapted to be executed by a Java Virtual Machine (JVM) running on the Web client's computer system. Servlet http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072 also includes static content, such as standard HTML or XML code. Resulting servlet http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072 is executed and writes a combination of static and dynamic text to the response object. That text, in turn, is returned to the STB client as response. STB client receives response and the resulting dynamic and static content is displayed using the client's browser software and a Java Virtual Machine on the client's computer used to

process the dynamic content. In this embodiment of the invention, the STB client hyperlinking step causes servlet http://PrintHD.TV/text-2/ad/ad2.asp?ID=7072 to print the text to the STB client to display pre-defined printable content on STB client screen.

In this embodiment, the first attribute identifying predetermined hyperlink address to predetermined hyperlinked content may be a resource identifier to identify a resource related to a Java Virtual Machine (JVM) and an application executed in the JVM, said resource identified by a resource management system comprising: a resource consumer management unit which generates a resource consumer for each of threads, wherein the resource consumer uses a resource; and a resource allocation policy manager which defines a resource identifier enabling an arbitrary resource to be uniquely identified, generates one or more resource manager for each resource identifier, wherein the at least one resource manager manages a resource, and sets a resource usage quota related to a resource identifier and a resource consumer, according to a request by the resource consumer management unit; wherein the resource manager calculates a quantity of resource usage by resource consumers generated for all applications and threads in the applications, stores the calculated quantity of resource usage, and informs about the quantity of resource usage wherein the resource identifier and the resource manager are generated by using Java language, said resource identifier comprises a unique identifier related to an application that is to be executed; storing the unique identifier; setting a resource allocation policy calling a resource identifier that is in an initial array position of a list in which resource identifiers uniquely identifying resources are arrayed; if the resource identifier exists in the initial array position, generating a resource manager related to the resource identifier that is a resource, setting a maximum usage value that is a quota related to the generated resource manager, registering the resource manager in a resource manager registry in which registered resource managers are stored according to types, and storing information on the generated resource manager and application identifier; and starting an application, and changing a state of the application according to a lifecycle of the application, said resource identified from a plurality of resources of threads by treating all of the threads performed in a Java Virtual Machine (JVM) as resource consumers, the method comprising: generating modified JVM threads related to the threads; wherein the generating the modified JVM threads comprises: calling a start( ) method of a certain thread object; requesting resource consumer object allocation for the certain thread object; generating a new resource consumer object according to the request, and obtaining an identifier of an application in which the certain thread object belongs; transferring information on the generated new resource consumer object and the application identifier, and requesting setting of a resource allocation policy for a corresponding resource consumer; and if the setting of the resource allocation policy for the corresponding resource consumer is ended, performing a run( ) function of the thread object and generating a native thread, said resource identified by a resource allocation policy comprising: requesting an initial resource identifier that is first on an array in a resource identifiers list; if a position for the initial resource identifier is not empty, having an identifier of an application to which a resource consumer related to a resource belongs, and obtaining a resource manager allocated to the resource; registering a resource consumer in the obtained resource manager; and setting a maximum value that may be used by the resource consumer.

Said predetermined hyperlink address is for a request for dynamically generated information from a client computer of a client-server network designating a thread to handle said request mapped, with said thread, said request to a selected servlet object of a plurality of instantiated servlet objects to invoke said selected servlet object to dynamically generate information to pass said dynamically generated information to said client computer wherein said mapping step includes the step of mapping said request to a selected servlet on a remote server computer wherein said invoking step includes the step of invoking said selected servlet object from said remote server computer to dynamically generate information.

Further, said predetermined hyperlink address may comprise a request from a client computer of said client-server network, said request requiring dynamically generated information from a servlet object of said client-server network to upload from a remote server computer of said client-server network a specified servlet object corresponding to said request and execute said specified servlet object to obtain dynamically generated information corresponding to said request wherein said specified servlet object and an application program interface are specified as object bytecodes in the JAVA programming language to send said dynamically generated information from said web server to said client computer wherein said local server computer stores a plurality of servlet objects, each of said servlet objects continuously operating until invoked in response to a specified request from a client computer wherein said plurality of servlet objects pass data to one another, said data passed may comprise one or more parameters indicated in said second attribute defining said pre-defined printable output of said predetermined hyperlinked content wherein selected servlet objects of said plurality of servlet objects are instantiated in response to said first attribute indicating said predetermined hyperlink address comprising an activated servlet URL wherein said second attribute one or more parameters comprises servlet HTML arguments including PrintStream out=new PrintStream(res-.getOutputStream( )); out.println("<html>"); out.println, parameters which were passed through standard HTML forms, including the HTTP method to be performed and the URI, which identifies the destination of the request, said predetermined hyperlink address comprising a request for dynamically generated information from a servlet object of said server computer indicating a servlet URL corresponding to said request, said servlet URL comprising arguments including said one or more parameters identifying said predefined printable output of said predetermined hyperlinked content indicated in said second attribute, said predetermined hyperlink address to a specified servlet object to obtain dynamically generated information corresponding to said request to pass said dynamically generated information to said client computer from a remote server computer storing a set of servlet objects that can be passed to said server computer storing a plurality of servlet objects, each of said servlet objects continuously operating until invoked in response to a specified request from said client computer.

Clearly, numerous modifications and variations of the instant invention are possible in light of the above teachings. It is therefore understood that, within the scope and spirit of the claims made herein, the invention may be practiced otherwise than as specifically described herein and the invention may be modified in arrangement and detail without departing from such scope and spirit.

What is claimed is:

1. A computer-implemented system having a processor for processing an hyperlink address string in conjunction with program signals representative of predetermined program